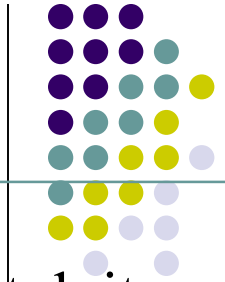
Schéma d’un document XML

Manipulation de données XML

El hadji Mamadou NGUER Enseignant chercheur en Informatique à l’UVS

Séquence 2 : Schéma d'un document XML

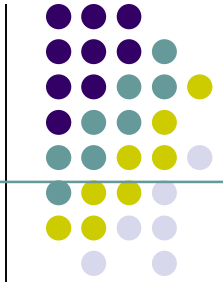


Objectifs spécifiques : A la suite de cette séquence, l'apprenant doit être capable de:

- Définir un schéma d'un document XML avec XML SCHEMA
- Comparer les technologies DTD et XML SCHEMA

elhadjimamadou.nguer@uvs.edu.sn

Séquence 2 : Schéma d'un document

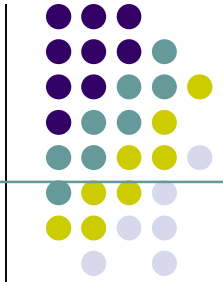


Plan :

1. Introduction
2. Schéma d'un document XML
3. Les technologies de définition d'un schéma
4. Définition d'un schéma XML avec XML SCHEMA

elhadjimamadou.nguer@uvs.edu.sn

Introduction



- Un document HTML peut être affiché par une application (un navigateur web par exemple) même si elle contient des erreurs (comme l'absence de balise de fermeture).

- Ce qui n'est pas le cas pour un document XML qui arrêterait le programme car XML ne tolère aucune erreur. .

- Cela est dû à la spécification XML du W3C qui indique le programme traitant un document XML doit cesser de fonctionner s'il en constate une erreur. La raison est qu'un logiciel traitant du XML doit être petit, rapide et compatible.

- Ainsi un document XML doit être **bien formé**.

elhadjimamadou.nguer@uvs.edu.sn

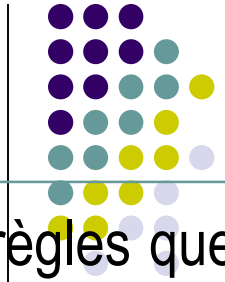
Introduction

Document XML bien formé (Rappel)

- Un document XML ayant un syntaxe correct est dit “bien formé". C’est-à-dire que un document :
  - avec un seul élément racine,
  - et toutes ses balises fermées
  - où les éléments sont correctement imbriqués
  - et les balises sont sensibles à la casse
  - et dont les attributs sont cités.
- Cette condition garantie le traitement d’un document XML par un programme mais ne permet pas de contrôler son contenu.

elhadjimamadou.nguer@uvs.edu.sn

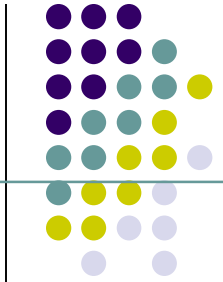
Schéma d'un document XML



- Le schéma (définition) d'un document XML est un ensemble de règles que l'on impose au document.
- Ces règles permettent de décrire la façon dont le document XML doit être construit.
- Elles peuvent être de natures différentes. Par exemple, ces règles peuvent imposer :
  - le nombre d'occurrence d'un élément,
  - la présence d'un attribut,
  - l'ordre d'apparition des éléments dans le document,
  - le type d'une donnée (nombre entier, chaîne de caractères, etc.).
- Un document XML bien formé et conforme à un schéma est dit "**valide**". Cela signifie que le document XML respecte toutes les règles qui lui sont imposées dans le schéma.

elhadjimamadou.nguer@uvs.edu.sn

Schéma d'un document XML



Quand utiliser un schéma?

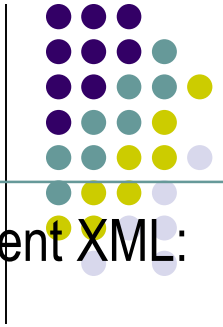
- Pour permettre à des personnes indépendantes d'échanger des données.
- Pour vérifier la validité des données reçues.
- Pour vérifier ses propres données.

Quand ne pas utiliser un schéma?

- Lorsqu'on fait des testes sur XML ou lorsqu'on travaille avec de petits fichiers XML, un schéma peut constitue une perte de temps.
- Pendant le développement d'une application pour ne pas que les erreurs de validation bloquent son fonctionnement. Il est préférable d'attendre que la spécification soit stable pour ajouter la définition du document.

elhadjimamadou.nguer@uvs.edu.sn

Technologies de definition d'un schéma



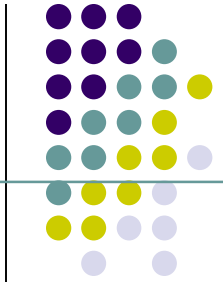
Il existe plusieurs technologies permettant de définir le schéma d'un document XML:

- **DTD (Document Type Definition) :** Définition de Type de Document
  - Elle est la première technologie élaborée pour définir un schéma XML.
  - C'est une recommandation de W3 et sa dernière version 2.1 date de 1998.
- **XML Schema :**
  - C'est une alternative au DTD basée sur XML
  - Il est publié comme recommandation par le W3C en mai 2001
  - Il est à la version 1.0 dont la dernière édition date de 2004
- **Relax NG, NVDL, Schematron, etc.**

Nous utilisons DTD et XML SCHEMA qui sont des recommendations de W3C

elhadjimamadou.nguer@uvs.edu.sn

Définition d'un schema avec XSD

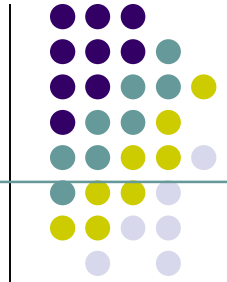


Introduction

- XSD est une recommandation du W3C qui précise la manière de décrire formellement les éléments XML.
- C'est une alternative au DTD basée sur XML et palie aux limites de ce dernier
- Il propose, en plus des fonctionnalités fournies par DTD, des nouveautés telles que :
  - l'utilisation de la **syntaxe XML**
  - le **typage des données** : de nombreux types sont intégrés (comme les booléens, les entiers, les réels et dates...) avec possibilité d'ajout de contraintes sur les types
  - la possibilité de **créer ses propres types de données**
  - la **notion d'héritage**. Les éléments peuvent hériter d'autres éléments.
  - La prise en charge des **espaces de nom**.
  - la **facilité de conception** et la **modularité** des schémas.

elhadjimamadou.nguer@uvs.edu.sn

Définition d'un schema avec XSD



Exemple

L'exemple suivant est un fichier XML Schema appelé "note.xsd" qui définit les éléments du document XML ci-dessus ("note.xml"):

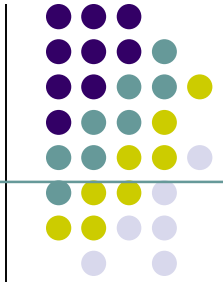
```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ugb.sn"  xmlns="http://www.ugb.sn"
elementFormDefault="qualified">
<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="a" type="xs:string"/>
      <xs:element name="de" type="xs:string"/>
      <xs:element name="titre" type="xs:string"/>
      <xs:element name="corps" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Document XML

```
<?xml version="1.0"?>
<note>
  <a>Caam</a>
  <de>Yaaya</de>
  <titre>Rappel</titre>
  <corps>
      Reunion à 12h
  </corps>
</note>
```

Définition d'un schema avec XSD

Exemple : (Explication)



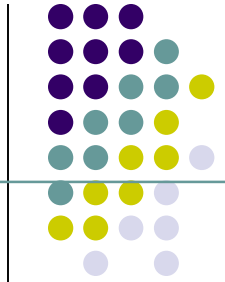
- L'élément note est de **type complexe** car il contient d'autres éléments.
- Les autres éléments (a, de, titre, corps) sont de **type simple** car ils ne contiennent pas d'autres éléments.
- **Référence au document XSD**

Ce document XML à une référence à un schéma XML:

```
<?xml version="1.0"?>
<note xmlns="http://www.ugb.sn"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ugb.sn note.xsd">

  <a>Caane</a>
  <de>Aysatu</de>
  <titre>Reminder</titre>
  <corps>Don't forget the wedding!</corps>
</note>
```

Définition d'un schema avec XSD



L'élément <schema>

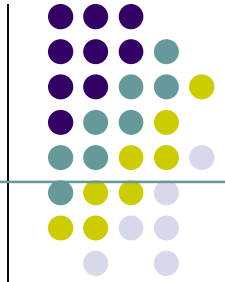
- C'est l'élément racine de chaque schéma XML:

```
<?xml version="1.0"?>
<xs:schema>
...
</xs:schema>
```

- Il peut contenir certains attributs. Une déclaration de schéma ressemble souvent quelque chose comme ceci:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ugb.sn"
xmlns="http://www.ugb.sn"
elementFormDefault="qualified">
...
</xs:schema>
```

Définition d'un schema avec XSD



L'élément <schema>

- Le fragment de code suivant:

xmlns:xs="http://www.w3.org/2001/XMLSchema"

indique que les éléments et les types de données utilisés dans le schéma proviennent du namespace "http://www.w3.org/2001/XMLSchema".

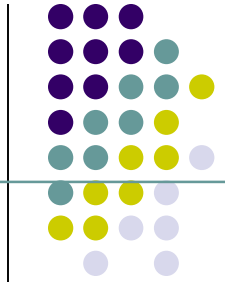
Il précise également que les éléments et les types de données venant de l'espace de noms "http://www.w3.org/2001/XMLSchema" sont préfixées avec **xs:**

- Le fragment de code suivant:

targetNamespace="http://www.ugb.sn"

indique que les éléments définis par ce schéma (note, to, from, heading, body.) proviennent de l'espace de noms "http://www.ugb.sn".

Définition d'un schema avec XSD



L'élément <schema>

- Le fragment de code :

xmlns="http://www.ugb.sn"

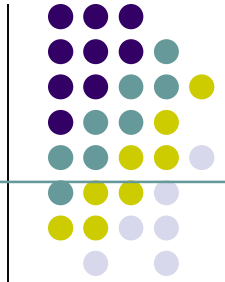
indique que l'espace de nom par défaut est "http://www.ugb.sn".

- Le fragment de code :

elementFormDefault="qualified"

indique que tous les éléments utilisés par le document XML qui sont déclarés dans ce schéma doivent être de namespace qualifié. C'est-à-dire que les éléments et les attributs sont dans le namespace cible (targetNamespace) du schéma.

Définition d'un schema avec XSD



L'élément <schema>

- **Référencer un schéma dans un document XML**

Le fragment de code : **xmlns="http://www.ugb.sn"**

spécifie la déclaration d'espace de noms par défaut. Elle indique que tous les éléments utilisés dans ce document XML sont déclarés dans le namespace "http://www.ugb.sn".

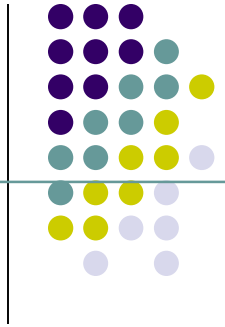
Une fois le namespace de l'instance du schéma XML est disponible:

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

On peut utiliser l'attribut schemaLocation. Cet attribute a deux valeurs, séparatées par un espace. La premiere est le namespace à utilizer et la seconde l'URL du schéma XML à utiliser pour ce namespace:
xsi:schemaLocation="http://www.ugb.sn note.xsd"

Les éléments simples

Qu’est ce qu’un élément simple?



- Un élément simple est un élément XML qui peut contenir seulement du texte. Il ne peut pas contenir d'autres éléments ou attributs.

- Cependant, la restriction "texte seulement" est tout à fait trompeur. Le texte peut être de plusieurs types différents. Il peut être l'un des types inclus dans la définition de schéma XML (boolean, string, date, etc.), ou un type personnalisé défini soi-même.

- On peut également ajouter des restrictions à un type de données afin de limiter son contenu, ou exiger que les données correspondent à un modèle spécifique.

Les éléments simples

Définition d'un élément simple

La syntaxe pour définir un élément simple est:

```
<xs:element name="xxx" type="yyy"/>
```

où xxx est le nom de l'élément et yyy est le type de la valeur de l'élément.

XML Schema a beaucoup de types de données intégrés dont les plus courants sont:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Les éléments simples

Définition d'un élément simple

Exemple

Voici quelques éléments XML:

```
<lastname>Xaadim</lastname>
<age>5</age>
<dateborn>2011-11-27</dateborn>
```

Et les définitions correspondantes:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Les éléments simples

Valeur par défaut et valeur fixe d'un élément simple

- Un élément simple peut avoir une valeur par défaut ou une valeur fixe spécifiée.

- Une valeur par défaut est automatiquement attribuée à l'élément quand aucune autre valeur n'est spécifiée.

Exemple : Dans l'exemple suivant, la valeur par défaut est "red":

```
<xs:element name="color" type="xs:string" default="red"/>
```

- Une valeur fixe est automatiquement assignée à l'élément et on ne peut pas spécifier une autre valeur..

Exemple : Dans cet exemple, la valeur fixe est "red":

```
<xs:element name="color" type="xs:string" fixed="red"/>
```

Les attributs

Element simple attribut?

Un élément simple ne peut avoir d'attribut. Si un élément a des attributs, il est considéré comme étant de type complexe. Mais l'attribut lui-même est toujours déclaré comme un type simple.

Comment définir un attribut?

La syntaxe pour définir un attribut est:

```
<xs:attribute name="xxx" type="yyy"/>
```

où xxx est le nom de l'attribut et yyy le type de la valeur de l'attribut.

Exemple : Voici un élément XML avec un attribut:

```
<lastname lang="EN">Obama</lastname>
```

Et voici la définition de l'attribut correspondant:

```
<xs:attribute name="lang" type="xs:string"/>
```

Les attributs

Valeur par défaut et valeur fixe des attributs

- Un attribut peut avoir une valeur par défaut ou une valeur fixe spécifiée.
- Une valeur par défaut est automatiquement affectée à l'attribut quand aucune autre valeur n’est spécifiée.

Dans cet exemple, la valeur par défaut est "EN":

```
<xs:attribute name="lang" type="xs:string" default="EN"/>
```

- Une valeur fixe est automatiquement affectée à l'attribut, et on ne peut pas spécifier une autre valeur.

Dans cet exemple, la valeur fixe est "EN":

```
<xs:attribute name="lang" type="xs:string" fixed="EN"/>
```

Les attributs

Attribut facultatif et attribut obligatoire

Un attribut est facultatif par défaut. Pour spécifier qu'il est requis, on utilise attribut "use":

```
<xs:attribute name="lang" type="xs:string" use="required"/>
```

Restrictions sur les contenus

- Quand un élément XML ou attribut a un type de données défini, il met des restrictions sur le contenu de l'élément ou attribut.

- Si un élément XML est de type "xs: date" et contient une chaîne comme "Bonjour tout le monde", l'élément ne sera pas valide.

- Avec les schémas XML, on peut également ajouter ses propres restrictions à ses éléments et attributs XML. Ces restrictions sont appelées facettes. On en saura plus sur les facettes dans la suite.

Les restrictions

Une restriction est utilisée pour définir des valeurs acceptables pour les éléments ou les attributs XML. Les restrictions sur les éléments XML sont appelées facettes.

Restrictions sur les valeurs

L'exemple suivant définit un élément appelé «âge» avec une restriction. La valeur de l'âge ne peut pas être inférieure à 0 ou supérieure à 120:

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Les restrictions

Restriction sur un ensemble de valeurs

Pour limiter le contenu d'un élément XML à un ensemble de valeurs, on doit utiliser la contrainte d'énumération.

L'exemple ci-dessous définit un élément appelé "voiture" avec une restriction. Les seules valeurs acceptables sont: Audi, Golf, BMW:

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Les restrictions

Restrictions sur un ensemble de valeurs

L'exemple ci-dessus pourrait également être écrit comme suit:

```
<xs:element name="car" type="carType"/>

<xs:simpleType name="carType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Audi"/>
    <xs:enumeration value="Golf"/>
    <xs:enumeration value="BMW"/>
  </xs:restriction>
</xs:simpleType>
```

Remarque: Dans ce cas, le type "carType" peut être utilisé par d'autres éléments, car il ne fait pas partie de l'élément "voiture".

Les restrictions

Restrictions sur une série de valeurs

- Afin de limiter le contenu d'un élément XML à une série de chiffres ou de lettres qui peuvent être utilisés, on utilise la contrainte pattern.

- L'exemple ci-dessous définit un élément appelé "lettre" avec une restriction. La seule valeur acceptable est l'une des lettres en MINUSCULES de a à z:

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Les restrictions

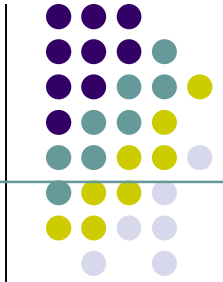
Restrictions sur une série de valeurs

L'exemple suivant définit un élément appelé "initials" avec une restriction. La seule valeur acceptable est trois des lettres MAJUSCULES de A à Z:

```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][A-Z][A-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Les restrictions

Restrictions sur une série de valeurs



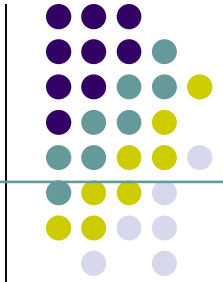
- L'exemple suivant définit également un élément appelé "initials" avec une restriction.

- La seule valeur acceptable est trois lettres majuscules ou minuscules de a à z:

```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Les restrictions

Restrictions sur une série de valeurs



- L'exemple suivant définit un élément appelé «choix» avec une restriction.

- La seule valeur acceptable est l'une des lettres suivantes: x, y, ou z

```
<xs:element name="choice">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[xyz]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Les restrictions

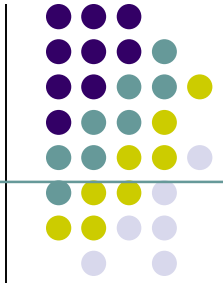
Restrictions sur une série de valeurs

- L'exemple suivant définit un élément appelé "prodid" avec une restriction.

- Les seules valeurs possibles sont les nombres de CINQ chiffres compris entre 0 à 9:

```
<xs:element name="prodid">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Les éléments complexes



Qu'est-ce qu'un élément complexe?

C’est un élément XML qui contient d'autres éléments et/ou des attributs.
Il existe quatre types d'éléments complexes:

- les éléments vides
- les éléments ne contenant que d'autres éléments
- les éléments ne contenant que du texte
- les éléments contenant à la fois des éléments et du texte

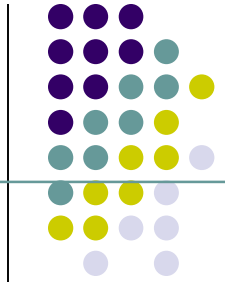
Note: Chacun de ces éléments peut contenir des attributs aussi bien!

Exemple :

Un élément complexe, "produit", qui est vide:

```
<product pid="1345"/>
```

Les éléments complexes



Exemple :

- Un élément complexe "employee" ne contenant que d'autres éléments:

```
<employee>
  <firstname>Jean</firstname>
  <lastname>Sarr</lastname>
</employee>
```

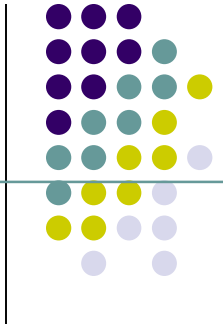
- Un élément complexe "food" ne contenant que du texte:

```
<food type="dessert">Ice cream</food>
```

- Un élément complexe "description" contenant à la fois des éléments et du texte:

```
<description>
It happened on <date lang="wolof">03/03/99</date> ....
</description>
```

Les éléments complexes



Déclaration d'un élément complexe vide

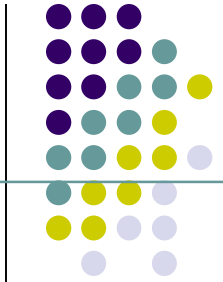
Un élément XML vide: <product prodid="1345" />

L'élément "product" ci-dessus n'a pas de contenu. Pour définir un type sans contenu, nous devons définir un type qui permet des éléments dans son contenu, mais sans y déclarons d'éléments, comme suit:

```
<xs:element name="product">
  <xs:complexType>
    <xs:attribute name="prodid" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

Les éléments complexes

Déclaration d'un élément complexe vide



On peut aussi donner à l'élément complexType un nom, et laisser l'élément "product" avoir un attribute type qui se réfère au nom du complexType (avec cette méthode, plusieurs éléments peuvent se référer à un même type complexe):

```
<xs:element name="product" type="prodtype"/>

<xs:complexType name="prodtype">
  <xs:attribute name="prodid" type="xs:positiveInteger"/>
</xs:complexType>
```

Les éléments complexes

Déclaration d'un élément contenant que des éléments

Avec l'élément, "person", qui ne contient que des éléments:

```
<person>
  <firstname>Charles</firstname>
  <lastname>Fay</lastname>
</person>
```

On peut définir l'élément "person" dans un schema comme suit:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Les éléments complexes

Déclaration d'un élément contenant que des éléments

- Notons que la balise <xs: sequence> spéficie que les éléments "prenom" et "nom" doivent apparaître dans cet ordre dans un élément "personne".

- On peut donner à l'élément complexType un nom, et laisser l'élément "personne" avoir un attribut type qui fait référence au nom du complexType (ainsi plusieurs éléments peuvent se référer au même type complexe):

```
<xs:element name="person" type="persontype"/>

<xs:complexType name="persontype">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Les éléments complexes

Déclaration d'un élément complexe contenant que du texte

Ce type d'élément ne contient que du contenu simple (texte et attribut), donc on ajoute un élément simpleContent autour du contenu. Lors de l'utilisation de contenu simple, on doit définir une extension ou une restriction dans l'élément simpleContent, comme suit :

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="basetype">
        ....
        ....
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

OR

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="basetype">
        ....
        ....
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Les éléments complexes

Déclaration d'un élément complexe contenant que du texte

Conseil: On utilise l'élément d'extension / de restriction pour étendre ou restreindre le type simple de base de l'élément.

Voici un exemple d'élément XML, "shoesize", qui contient du texte uniquement:: <shoesize country="Gambia">35</shoesize>

L'exemple suivant déclare un complexType, "shoesize". :

```
<xs:element name="shoesize">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="country" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Les éléments complexes

Déclaration d'élément complexe contenant que du texte

On pourrait aussi donner à l'élément complexType un nom, et laisser l'élément "shoesize" avoir un attribut type qui fait référence au nom de l'élément complexType (ainsi plusieurs éléments peuvent se référer au même type complexe):

```
<xs:element name="shoesize" type="shoetype"/>

<xs:complexType name="shoetype">
  <xs:simpleContent>
    <xs:extension base="xs:integer">
      <xs:attribute name="country" type="xs:string" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Les éléments complexes

Déclaration d'un élément complexes avec contenu mixte

Un élément de type complexe mixte peut contenir des attributs, des éléments et du texte.

L'élément "letter" suivant contient à la fois du texte et d'autres éléments:

```
<letter lang="en">
  Dear Mr.<name>Pierre Gomez</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

Les éléments complexes

Déclaration d'un élément complexes avec contenu mixte

Le schéma suivant déclare l'élément "letter":

```
<xs:element name="letter">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="orderid" type="xs:positiveInteger"/>
      <xs:element name="shipdate" type="xs:date"/>
    </xs:sequence>
     <xs:attribute name="lang" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Remarque: Pour permettre aux données caractères d'apparaître entre les éléments enfants de «letter», l'attribut mixed doit être à "true".

Les indicateurs

On peut contrôler la façon dont les éléments doivent être utilisés dans les documents avec des indicateurs. Il y a sept indicateurs:

Les indicateurs d'ordre:

- All
- Choice
- Sequence

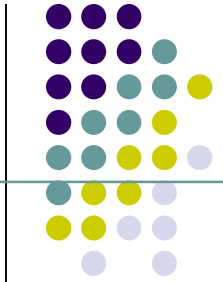
Les indicateurs d'occurrence :

- maxOccurs
- minOccurs

Les indicateurs de groupe :

- Group name
- attributeGroup name

Les indicateurs



Les indicators d'ordre

Les indicateurs d'ordre sont utilisés pour définir l'ordre des éléments.

- **L'indicateur All**

Il specifie que les éléments enfants peuvent apparaître dans n'importe quel ordre mais avec une seule occurrence à la fois:

```
<xs:element name="person">
  <xs:complexType>
    <xs:all>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Remarque: Lorsqu'on utilise l' indicateur <all> on peut régler l'indicateur <minOccurs> à 0 ou 1 et l'indicateur <maxOccurs> ne peut être défini qu'à 1 (Les indicateurs <minOccurs> et <maxOccurs> sont décrits plus loin).

Les indicateurs

Les indicators d'ordre

- **Indicateur Choice**

Il spécifie que c'est soit l'un soit l'autre des éléments enfant qui est choisi:

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Les indicateurs

Les indicators d'ordre

- **L'indicateur Sequence**

Il spécifie que les éléments enfant doivent apparaître dans un ordre spécifique:

```
<xs:element name="person">
  <xs:complexType>
   <xs:sequence>
     <xs:element name="firstname" type="xs:string"/>
     <xs:element name="lastname" type="xs:string"/>
   </xs:sequence>
 </xs:complexType>
</xs:element>
```

Les indicateurs

Les indicators d'occurence

Ils sont utilisés pour définir la fréquence d'apparition d'un élément.

Remarque: Pour les indicateurs d'ordre et de groupe la valeur par défaut pour maxOccurs et minOccurs est 1.

- **L'indicator maxOccurs**

L'indicateur <maxOccurs> indique le nombre maximum de fois qu'un élément apparait:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string" maxOccurs="10"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:element>
```

Les indicateurs

Les indicators d'occurence

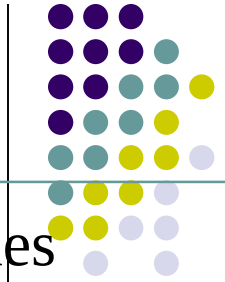
- **L'ndicateur minOccurs**

Il indique le nombre minimum de fois qu'un élément peut apparaître:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
      maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Remarque: Pour indiquer qu'un élément apparaisse un nombre illimité de fois, on utilise maxOccurs = "unbounded":

Les indicateurs



Les indicateurs de groupe : ils sont utilisés pour définir des ensembles d'éléments associés.

- **Groupe d'élements**

Des éléments sont regroupés avec la declaration group comme suit:

```
<xs:group name="groupname">
...
</xs:group>
```

On peut définir un élément all, choice ou sequence dans la déclaration du groupe.

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>
```

Les indicateurs

Les indicateurs de groupe :

Groupe d'élements: une fois qu'on définit un groupe, on peut en faire référence dans une autre définition, comme suit:

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:element name="person" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:group ref="persongroup"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Les indicateurs

Les indicateurs de groupe :

- **Groupe d'attributs**

Un groupe d'attributs est définis par la déclaration de attributeGroup, comme suit :

```
<xs:attributeGroup name="groupname">
...
</xs:attributeGroup>
```

L'exemple suivant définit un groupe d'attributs appelé "personattrgroup":

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>
```

Les indicateurs

Les indicateurs de groupe :

- **Groupe d'attributs**

Après avoir défini un groupe d'attributs, on peut le référencer dans une autre définition, comme suit:

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>

<xs:element name="person">
  <xs:complexType>
    <xs:attributeGroup ref="personattrgroup"/>
  </xs:complexType>
</xs:element>
```

Fin de la séquence 2